N 体シミュレーションの基礎

道越秀吾

国立天文台 CfCA

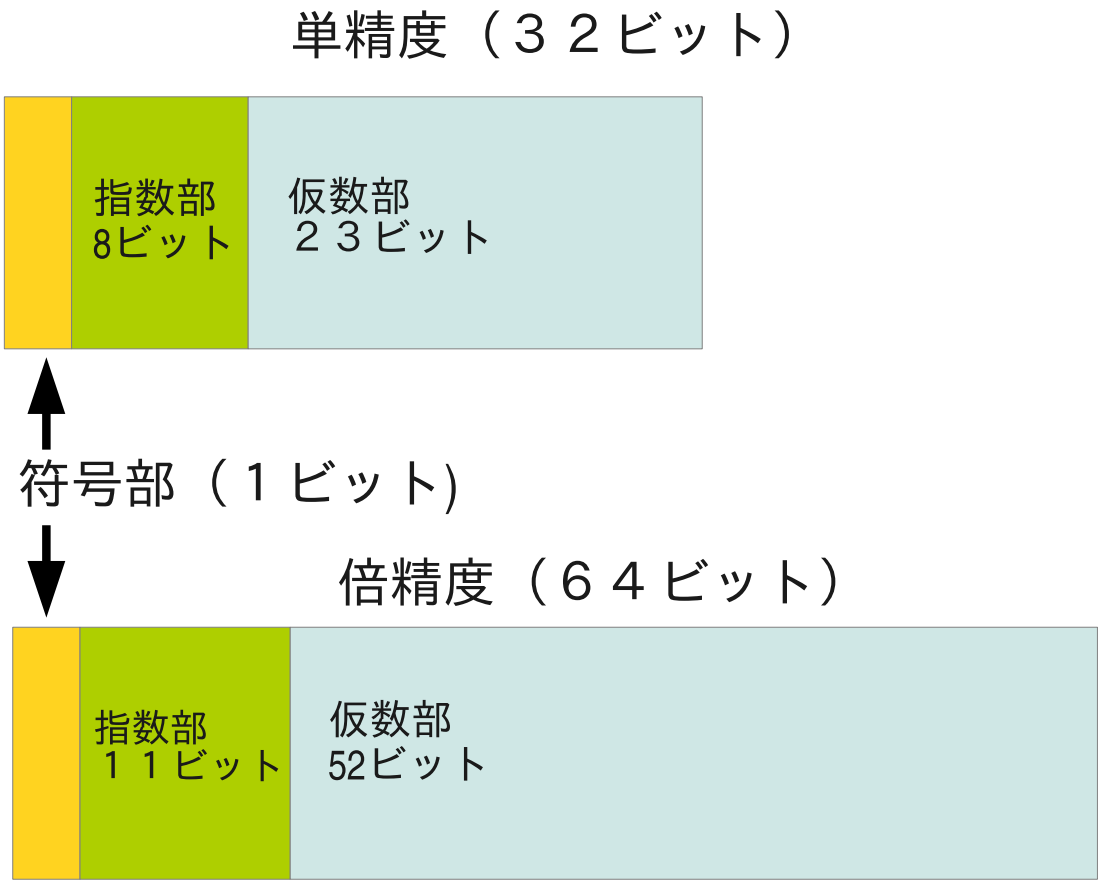
January 11, 2012

目次

目次

コンピュータによる実数の表現：浮動小数点数

- **実数はコンピュータでは浮動小数点数で表現される**
- **必要な計算精度に応じて単精度 (32 ビット) や倍精度 (64 ビット) が使われる**



	有効桁数
単精度	~ 7
倍精度	~ 15

$$\text{浮動少数} = \pm 2^{\text{指数部}} \times (1 + \text{仮数部})$$

誤差

丸め誤差

計算機内での実数表現によって生じる誤差。仮数分が有限であるので途中の桁で打ち切る（四捨五入等）。このときに生じる誤差のこと。

	丸め誤差
単精度	3×10^{-8}
倍精度	5×10^{-17}

打ち切り誤差

計算処理を続ければ正確な値が得られるのにも関わらず、途中で計算をやめることによって生じる誤差。例えば級数展開を途中で止めた場合。

$$\sin(x) = x - \frac{x^3}{3!} + \cdots = x - \frac{x^3}{3!} + \text{打ち切り誤差}$$

丸め誤差の発生：桁落ちと情報落ち

桁落ち

ほぼ等しい数の差を計算したときに有効桁数が下がること

$$1.234567(7\text{ 桁}) - 1.234566(7\text{ 桁}) = 1.0 \times 10^{-6}(1\text{ 桁})$$

情報落ち

大きな数に小さな数を足したり引いたりした場合、計算結果に反映されれないことによって生じる誤差

$$1.234567 - 1.234566 \times 10^{-6} = 1.234568$$

N 体シミュレーションの基礎方程式

基礎方程式

$$\frac{d\vec{v}_i}{dt} = \sum_{j \neq i}^{N} G m_j \frac{\vec{x}_j - \vec{x}_i}{|\vec{x}_j - \vec{x}_i|^3}$$

$$\frac{d\vec{x}_i}{dt} = \vec{v}_i$$

N 体シミュレーションのポイント

- 正確で高速に運動方程式を解くこと
- 重力の計算の量は粒子数の二乗に比例する ($O(N^2)$ = 定数 $\times N^2$)

- どうやって時間発展させるか？(今日の講義と講義 4)
- どうやって重力を高速に計算するか？(「講義 4：高度なシミュレーション法」ツリー法を扱う)

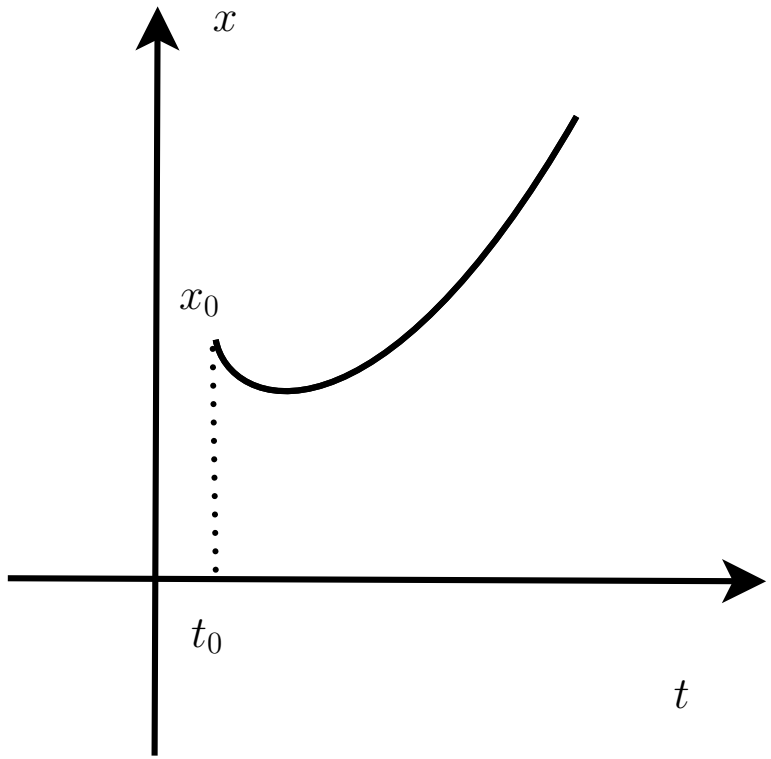
目次

常微分方程式

初期値問題

$$\frac{dx}{dt} = f(x,t)$$

$$x(t_0) = x_0$$



時刻 t_0 における x の値が x_0 として（初期条件）、任意の時刻 t における x を求める。

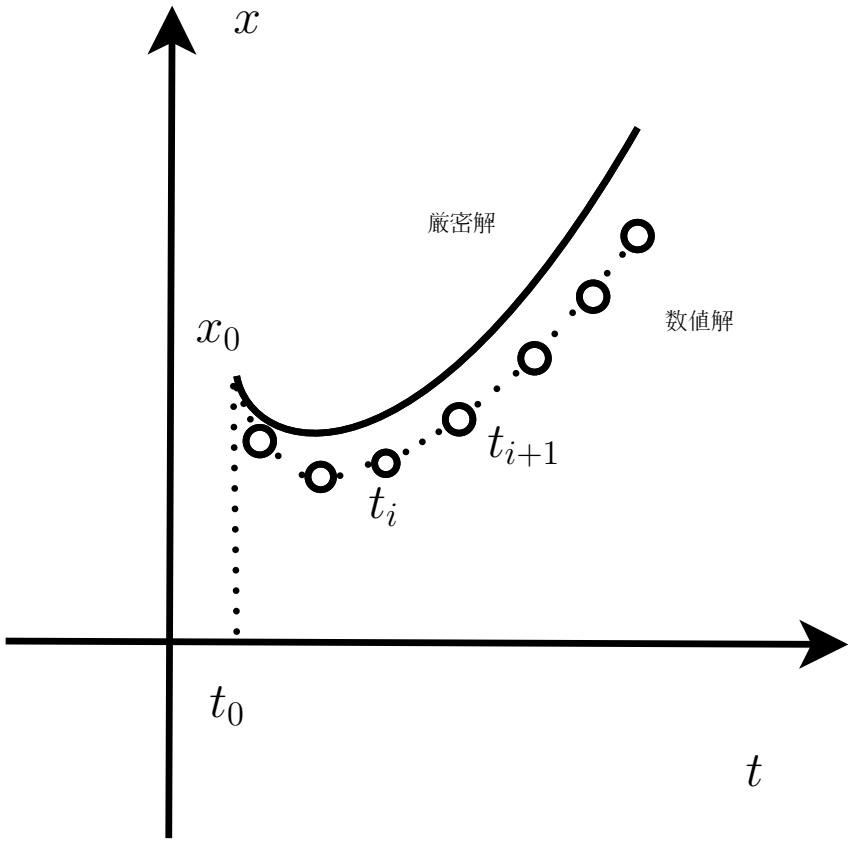
1

[1] 初期値問題以外では 2 点境界問題等。今日説明しない特別な解法を要する。

数値計算コードの骨格

プログラムの基本構造

```
1  void time_integration(
2    double *x, double t, double dt)
3  {
4    // ステップすすめる
5    ....
6  }
7  ...
8  t=t_start;
9  while (t<t_end) {
10   time_integration(&x,t,dt);
11   t += dt;
12 }
```

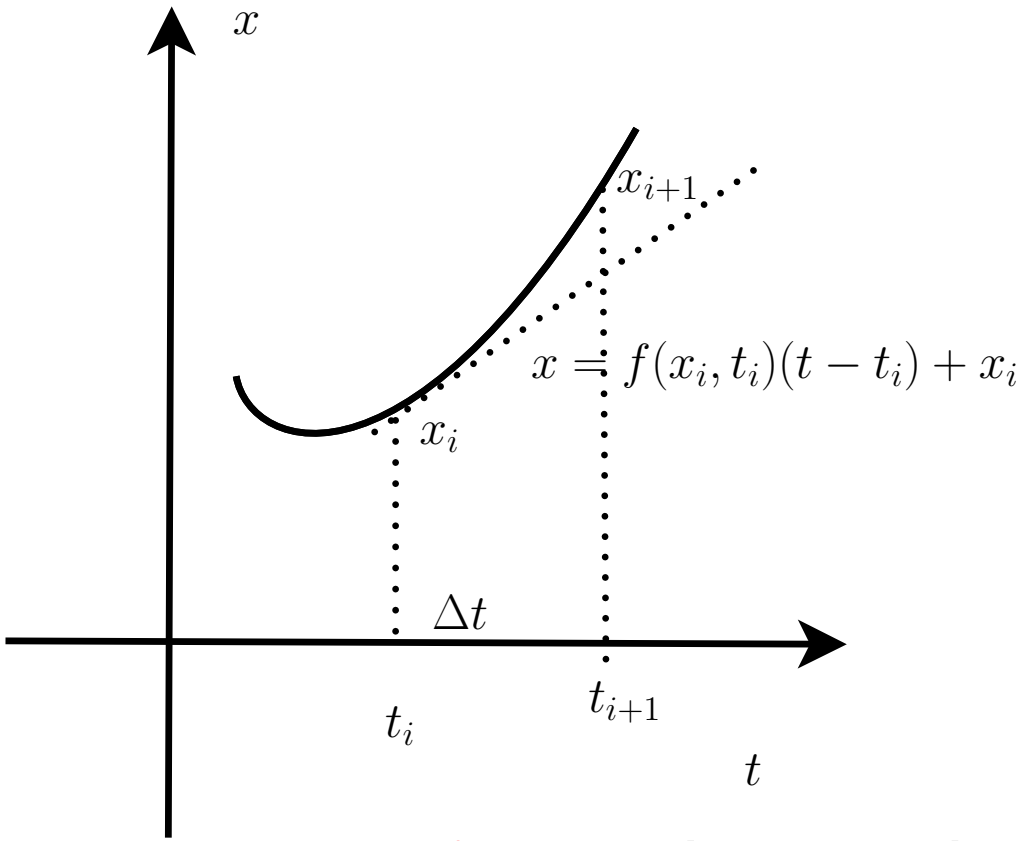


オイラー法

微分方程式

$$\frac{dx}{dt} = f(x,t)$$

を**接線近似**によって微小な時間 Δt 経過後の x を求める。



$$\frac{dx}{dt} \simeq \frac{x_{i+1} - x_i}{\Delta t} = f(x_i, t_i)$$

$$\Leftrightarrow x_{i+1} = x_i + \Delta t f(x_i, t_i)$$

Δt を**タイムステップ** (または時間刻み、時間ステップ) とよぶ。

オイラー法のプログラム例

$$\frac{dx}{dt} = f(x,t), \quad x_{i+1} = x_i + \Delta t f(x_i, t_i)$$

```
1  double calc_f(double x, double t)
2  {
3     // 右辺の計算f
4     ....
5  }
6  void time_integration( double *x, double t, double dt)
7  {
8    double f;
9     f = calc_f(*x, t);
10    *x = *x + dt * f;
11 }
12 ...
13 t=t_start;
14 while (t<t_end) {
15    time_integration(&x,t,dt);
16    t += dt;
17 }
```

オイラー法の計算例

$$\frac{d^2x}{dt^2} = -x$$

$$v_0 = 0,\ x_0 = 1$$

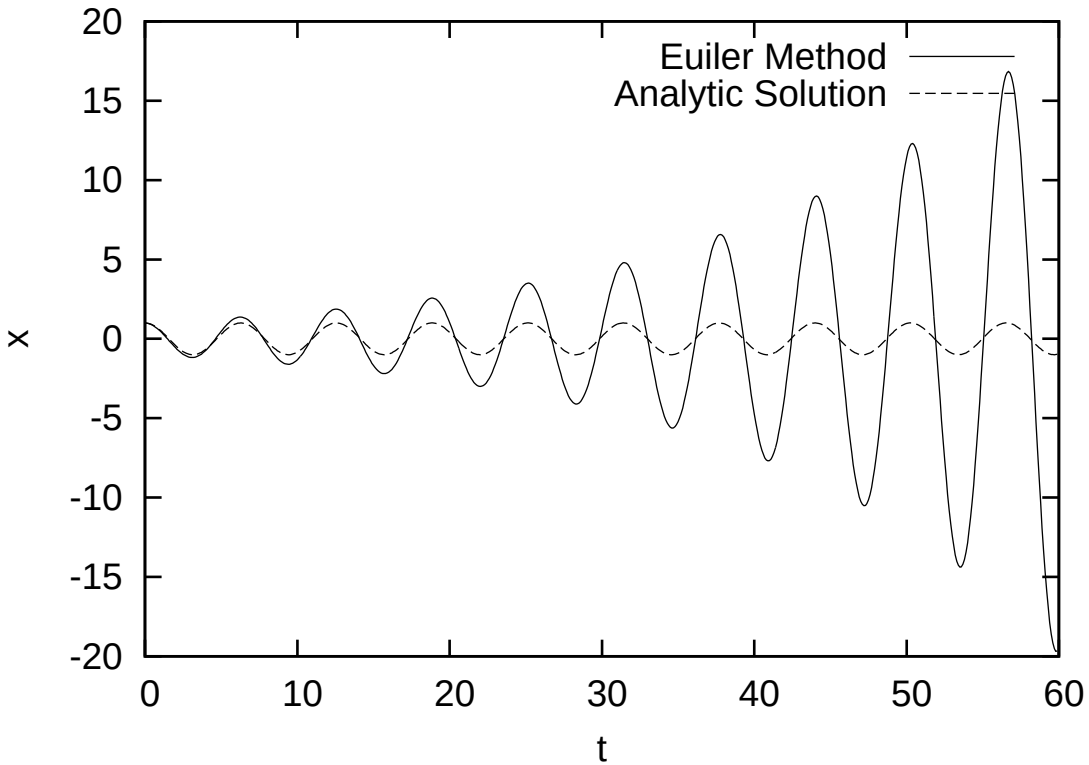
$\Longrightarrow$

厳密解

$$x = \cos(t)$$

この場合のオイラー法による数値解は、

$$\Delta t = 0.1$$



計算法の次数

- オイラー法はタイムステップ Δt が小さければ接線近似を行えることを基にしている
- したがって、この近似は、タイムステップ Δt が細かくなるほど良いはずである
- 誤差の大きさの振る舞いを定量的に表すには？

公式の次数

$x(t_{i+1})$ を厳密解、x_{i+1} を数値解とする。このとき

$$\text{誤差} = x(t_{i+1}) - x_{i+1} = O(\Delta t^{m+1}) = \text{定数} \times \Delta t^{m+1}$$

とかけると、この数値計算法は m 次であるという。

オイラー法の次数

テイラー展開をするとすぐに分かる

$$x(t_{i+1}) - x_{i+1} = O(\Delta t^2)$$

つまりオイラー法は 1 次である

大域誤差と局所誤差

- 前頁で紹介した誤差は、タイムステップ Δt の間に蓄積する誤差であった局所打ち切り誤差
- では一定の時間内で蓄積する誤差は？

大域打ち切り誤差

局所誤差が

$$\text{局所誤差} = x(t_{i+1}) - x_{i+1} = O(\Delta t^{m+1}) = \text{定数} \times \Delta t^{m+1}$$

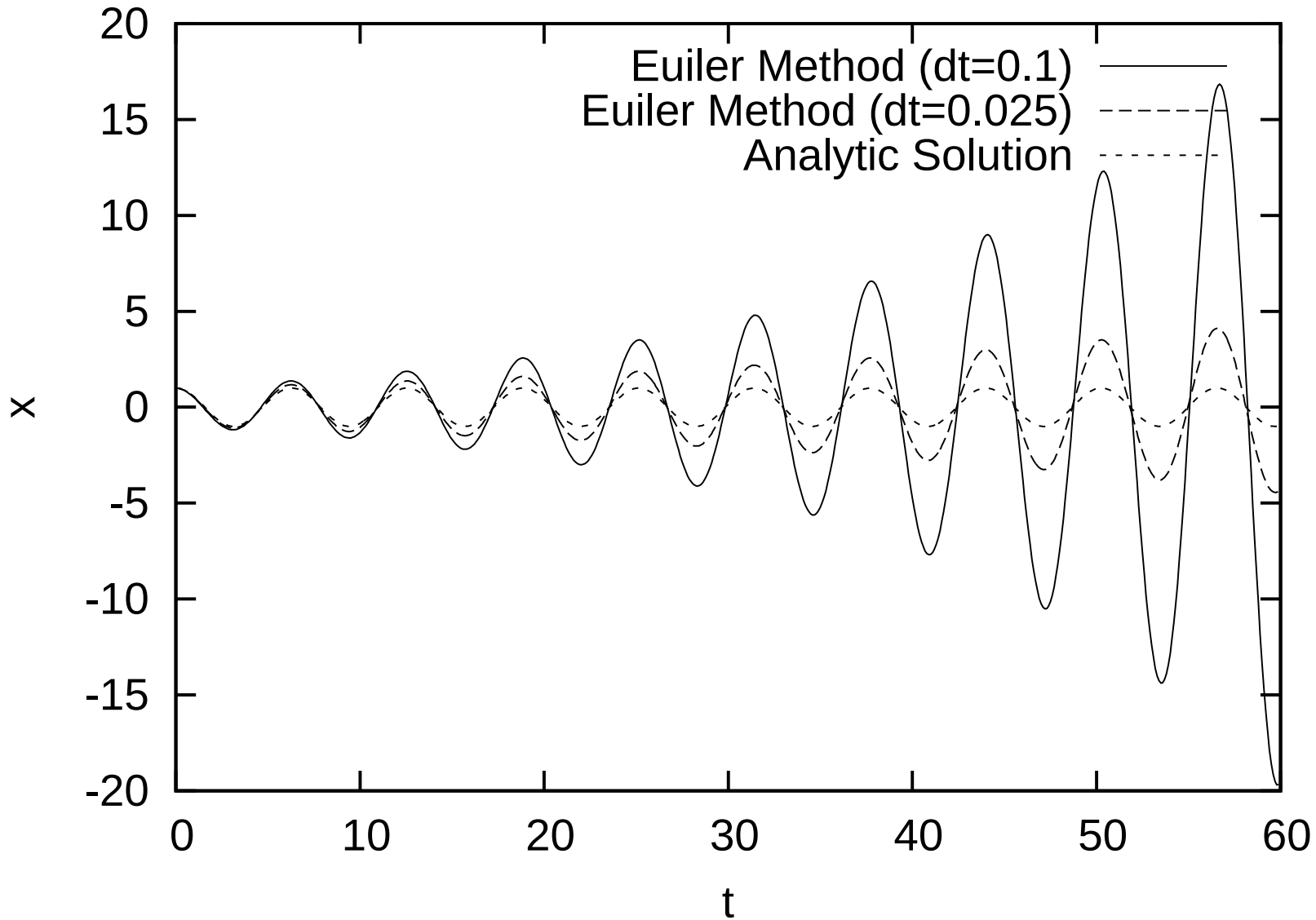
のとき大域誤差は

$$\text{大域誤差} = O(\Delta t^{m+1}) \times \frac{t}{\Delta t} = O(\Delta t^{m})$$

となる

したがって、タイムステップを細かくするほど大域誤差も小さくなる。

タイムステップを変えた場合



高精度計算の方法

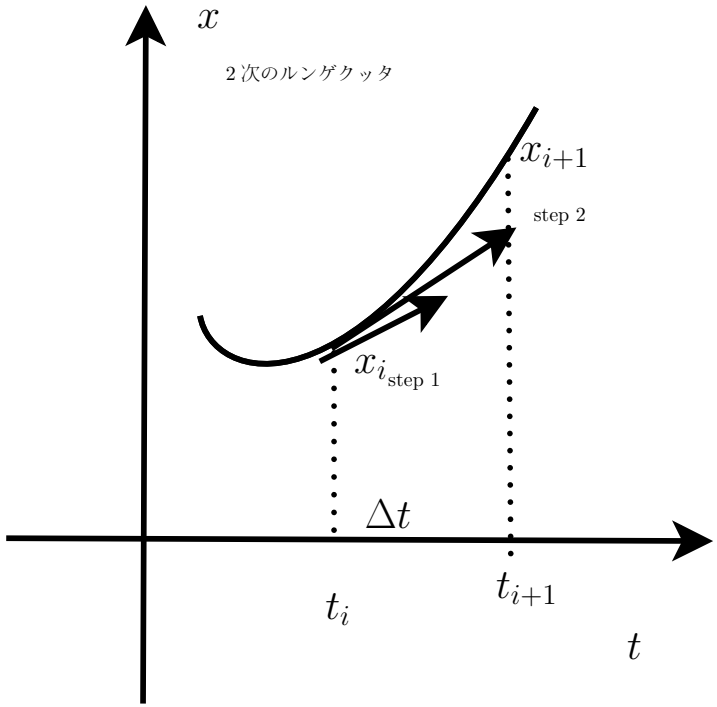
- タイムステップを小さくすれば良い
- しかし、タイムステップを小さくするほど計算量も増大する (計算回数 $= t/\Delta t$)
- そもそも、丸め誤差の影響でタイムステップを小さくしすぎると計算できない (情報落ち)

高速に正しい計算を行う方法は？

- 高次の解法 (ルンゲクッタ法、予測子修正子法、リチャードソン補外など)
- 考える系の構造や性質に着目する (後ほどハミルトン系の解法で詳しく説明)

2 次のルンゲクッタ法

ルンゲクッタ法は高次の解法の基本。この講義では簡単にその考え方と使い方を説明する。



- 半分のステップのところでの関数の値で微分を計算
- その値で本ステップを計算する

公式

$$k_1 = x_i + \frac{\Delta t}{2} f(x_i, t_i)$$

$$x_{i+1} = x_i + \Delta t f(k_1, t_i + \Delta t/2)$$

次数

この計算法の次数は 2 次。つまり

$$\text{局所誤差} = x(t_{i+1}) - x_{i+1} = O(\Delta t^3)$$

$$\text{大域誤差} = x(t) - x_N = O(\Delta t^2)$$

古典的ルンゲクッタ法

一般用途で最もよく用いられるのは 4 次のルンゲクッタ法

4 次のルンゲクッタ公式

$$k_1 = f(x_n, t_n)$$

$$k_2 = f(x_n + \Delta t k_1/2, t_n + \Delta t/2)$$

$$k_3 = f(x_n + \Delta t k_2/2, t_n + \Delta t/2)$$

$$k_4 = f(x_n + \Delta t k_3, t_n + \Delta t)$$

$$x_{n+1} = x_n + \Delta t \left(\frac{1}{6} k_1 + \frac{1}{3} k_2 + \frac{1}{3} k_3 + \frac{1}{6} k_4 \right)$$

- この公式は 4 次 [2]
- 古典的な数値計算法でよく使われる
- 係数が簡単で 4 次の公式だが計算量も少ない

[2] 一般のルンゲクッタ法についてはテキスト参照

高次のルンゲクッタ法

- 次数をあげるほど 1 ステップの誤差が減るが 1 ステップあたりの計算量も増える
- 5 次以上の陽的ルンゲクッタでは必要段数が増加する
- これらの兼ね合いで 4 次のルンゲクッタがよく用いられる

次数と必要な段数

段数	1	2	3	4	5	6	6	8	9	10
次数	1	2	3	4	4	5	6	7	7	7

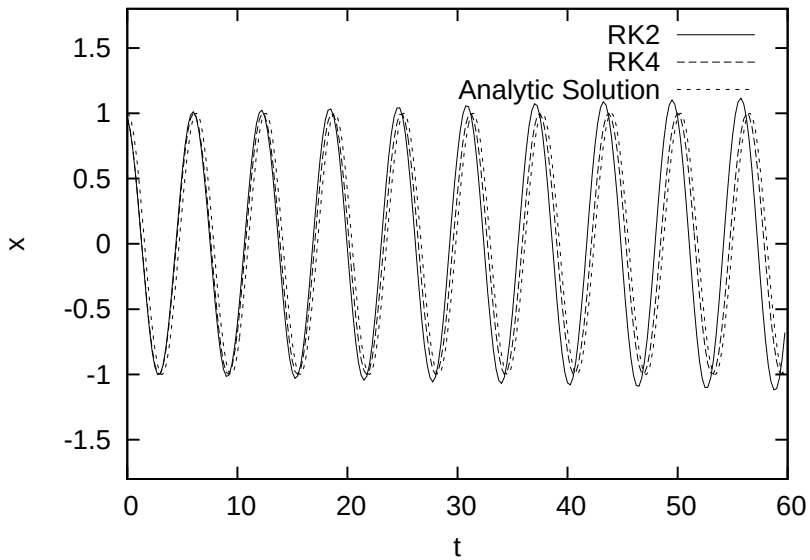
参考

5 次以上の公式に対しては、

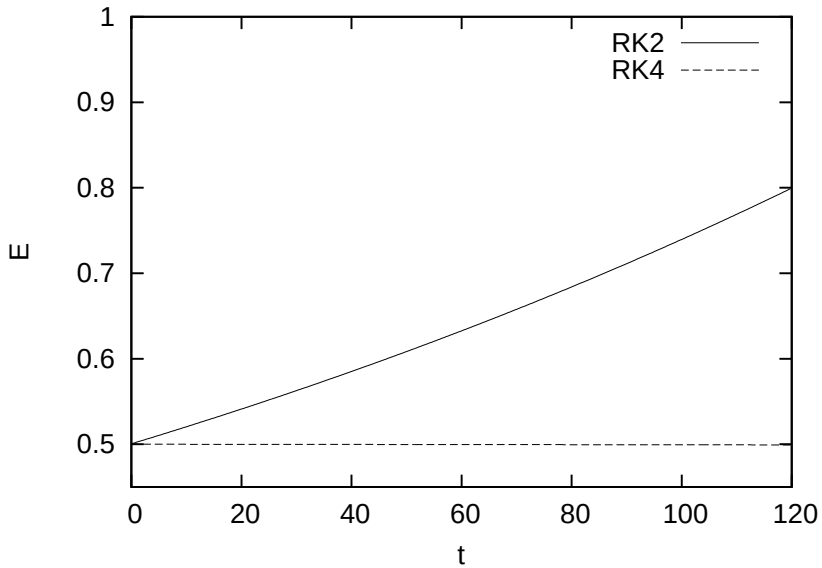
$$\text{次数} < \text{段数}$$

ルンゲクッタ法の計算例

数値解



エネルギー誤差



ルンゲクッタ法の性質

- オイラー法よりずっと**精度がよい**
- 次数を上げると同じタイムステップの場合誤差が**減少**する
- 時間がたつにつれて**誤差が大きく**なっていく

目次

N 体シミュレーションの解法

N 体シミュレーションの方法

- N 体シミュレーションも古典的ルンゲクッタ法で良いか？
- 実際の N 体シミュレーションでは別の方法がよく使われる
- 微分方程式の解をできるだけ近似するだけではなく、離散化した場合でも元の微分方程式のもつ数学的構造を再現する方がよい。

物理の方程式系ではエネルギーや角運動量等の保存量が存在する場合がある。このような保存量に関する望ましい性質を備える公式を考える。

N 体シミュレーションでの標準的な計算法

無衝突系

リープフロッグ公式

衝突系

エルミート公式 (「講義４:高度なシミュレーション法」で扱う)

リープフロッグ公式

リープフロッグ公式

$$v_{i+1/2} = v_i + a(x_i)\frac{\Delta t}{2}$$

$$x_{i+1} = x_i + v_{i+1/2}\Delta t$$

$$v_{i+1} = v_{i+1/2} + a(x_{i+1})\frac{\Delta t}{2}$$

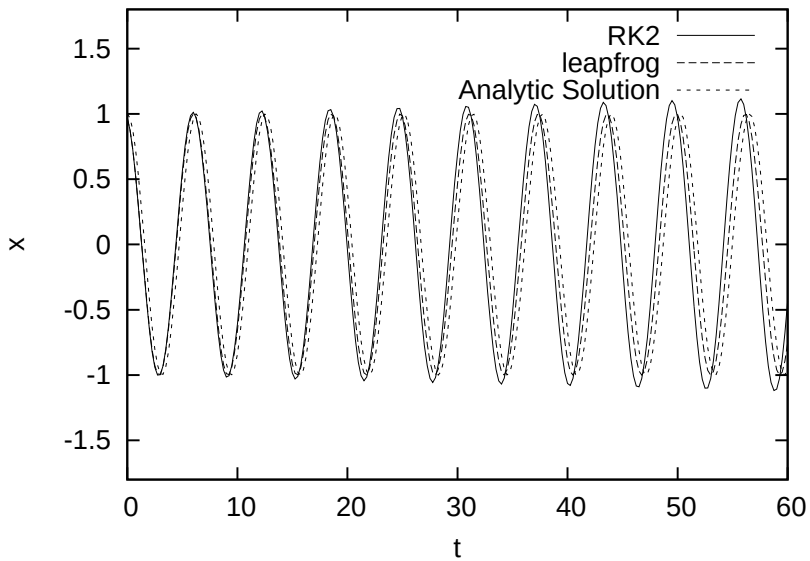
特徴

- 2 次の公式
- シンプレクティック公式
- 対称公式

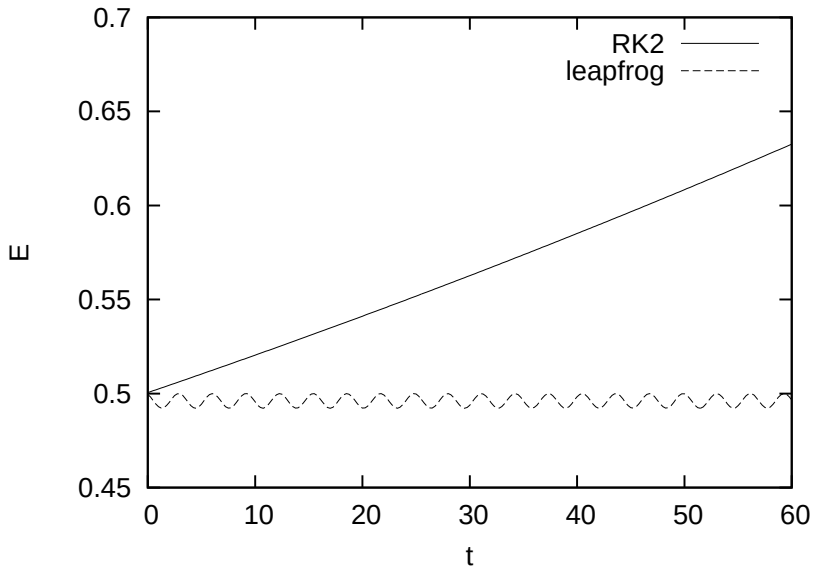
局所誤差はルンゲクッタに比べて良いわけではないが、良い性質をもつ

2 次のルンゲクッタとリープフロッグ公式

数値解



エネルギー誤差



リープフロッグ公式の性質

- リープフロッグ公式は長時間計算してもエネルギー誤差が溜まっていかない
- 今回の実習ではリープフロッグ公式を用いて計算する

リープフロッグ公式の実装の流れ

ここでは簡単のために１次元１粒子の場合。実習では３次元多粒子の場合をやる

```
1  void leapfrog(double *x, douvle *v,  double t, double dt)
2  {
3     // １ステップ分だけアップデートする
4     ....
5  }
6  ...
7  while (t<t_end) {
8     leapfrog(&x,&v,t,dt);
9     t += dt;
10 }
```

リープフロッグ公式のアルゴリズム

```
1  void calc_force(double *x)
2  {
3     // 加速度の計算
4     ....
5  }
6  void leapfrog( double *x, douvle *v,
7     double t, double dt)
8  {
9     // 1ステップ分だけアップデートする
10    ....
11 }
12 ...
13 while (t<t_end) {
14    leapfrog(&x,&v,t,dt);
15    t += dt;
16 }
```

リープフロッグ公式

$$v_{i+1/2} = v_i + a(x_i)\frac{\Delta t}{2}$$

$$x_{i+1} = x_i + v_{i+1/2}\Delta t$$

$$v_{i+1} = v_{i+1/2} + a(x_{i+1})\frac{\Delta t}{2}$$

アルゴリズム

1. v_0, a_0 から $v_{1/2}$ を計算する
2. $x_0, v_{1/2}$ から x_1 を計算する
3. x_1 から a_1 を計算する
4. $v_{1/2}, a_1$ から v_1 を計算する

重力の計算

```
1  void calc_force(int n,double m[]
2    double r[][3],...) {
3    for (i=0;i<n;i++) {
4      for (k=0;k<3;k++) {
5        a[i][k]=0.0;
6      }
7    }
8    for (i=0;i<n-1;i++) {
9      for (j=i+1;j<n;j++) {
10       a[i][k] += m[j]*...;
11       a[j][k] += m[i]*...;
12     }
13   }
14 }
```

重力の計算

$$\sum_{j \neq i}^{N} G m_j \frac{\vec{x}_j - \vec{x}_i}{(|\vec{x}_j - \vec{x}_i|^2 + \epsilon^2)^{3/2}}$$

- 計算量は、$O(N^2)$
- もっと賢い方法はツリー (講義 4)

今日の講義のまとめ

まとめ

- 数値計算法の基礎を説明した
  - 単精度、倍精度の意味
  - 誤差 (情報落ち、桁落ち)
- 微分方程式の解法の基本としてルンゲクッタ法を説明した。
  - 局所誤差と大域誤差
  - 公式の次数を定義した
  - オイラー法は１次
  - 古典ルンゲクッタは４次
- N 体シミュレーション用の解法としてリープフロッグ法を紹介した
  - リープフロッグ法は２次
  - ２次のルンゲクッタよりもエネルギー誤差が蓄積しない等の良い性質がある。
  - 今回の実習ではリープフロッグ法を使います